UCID-20255

THREE ALGORITHMS FOR PARAMETRIC FREQUENCY DOMAIN SYSTEM INDENTIFICATION

D. M. Tilly

November 1984

Lawrence Livermore National Laboratory

DISCLAIMER

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $4.50

Page Range	Domestic Price	Page Range	Domestic Price
001-025	$ 7.00	326-350	$ 26.50
026-050	8.50	351-375	28.00
051-075	10.00	376-400	29.50
076-100	11.50	401-426	31.00
101-125	13.00	427-450	32.50
126-150	14.50	451-475	34.00
151-175	16.00	476-500	35.50
176-200	17.50	501-525	37.00
201-225	19.00	526-550	38.50
226-250	20.50	551-575	40.00
251-275	22.00	576-600	41.50
276-300	23.50	601-up[1]	
301-325	25.00		

[1]Add 1.50 for each additional 25 page increment, or portion thereof from 601 pages up.

THREE ALGORITHMS FOR PARAMETRIC FREQUENCY DOMAIN SYSTEM IDENTIFICATION

Diane M. Tilly

CONTENTS

THREE ALGORITHMS FOR PARAMETRIC FREQUENCY DOMAIN SYSTEM IDENTIFICATION*

Diane M. Tilly

Lawrence Livermore National Laboratory
University of California
Livermore, CA 94550

ABSTRACT

System identification is used in many engineering applications. Most of these identification problems can be solved using one of the many time domain methods that have been studied and well developed in the last 15 years. However, in practice frequency response data, or Bode plots, may be the only data available to the engineer. Also, in some applications a time domain model of the system is not necessary. Instead, the engineer wishes to gain a general knowledge of system response. In these instances a frequency domain system identification technique may be more appropriate. For this particular project there was an interest in estimating the coefficients of a Laplace transform transfer function system model. The transfer function model would aid in the design of machine tool control systems. Several available parametric frequency domain identification methods were studied and three computer codes were chosen for further development as representatives of different approaches to the solution of the problem.

ACKNOWLEDGMENTS

I would like to thank Greg Clark for his guidance throughout this project. Also Chris Fairley for the helpful discussions, and the time spent in collecting data. Finally, I'd like to thank Freddie Barnes, Jim Candy, Don Gavel, Charlie Herget, and Vince McGevna for the many technical and practical discussions.

* Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

1.0 INTRODUCTION

System identification is a major part of many engineering problems. The identification problem involves finding a mathematical model of a system from measured input-output data [4]. The system model can be either non-parametric, or parametric. Often a non-parametric model, such as an impulse response or a simple Bode plot of the system response, is enough to satisfy the needs of the engineer. In other cases a parametric model, such as a difference equation or a Laplace transform transfer function, where the coefficients of the equation are the parameters identified, is necessary.

Historically, non-parametric frequency domain methods dominated practice of system identification in contol engineering applications up to the 1960's. In the late sixties, interest in time domain identification methods began to increase, and time domain techniques have been studied and developed for the last two decades. Most system identification problems can be solved using one of the many time domain algorithms available. However, there are some applications where a frequency domain identification technique is more appropriate. The purpose of this report is to compare three parametric frequency domain techniques and discuss their advantages and disadvantages.

One important consideration when deciding on a system identification technique is the system model and its use. For instance, if the model is to be used to simulate the system, or used in any state-space control system design procedure, then time domain techniques are best. However, many engineers are more familiar with frequency response methods and gain a more intuitive understanding of the system from frequency response plots. If frequency response control system design methods are used, or if the model is used to gain general insight into the system, such as resonances in the response, then frequency domain methods should be considered.

Another important consideration is the type of data available. Since the development of the fast and inexpensive Fast Fourier Transform algorithm many data aquisition and analysis systems immediately transform the data to the frequency domain. In other instances the only data available is a Bode plot of the system transfer function, or frequency response data from a swept sine source, where no time domain data exists. The engineer is then limited to frequency response data, and frequency domain identification methods are required.

Here we are interested in identifying poles and zeros of a Laplace transform system transfer function. The transfer function model would then be used to aid in the design of machine tool control systems. The only data available was the auto- and cross-spectra of the input (x) and output (y) records of the system, $Sxx(f)$ and $Sxy(f)$, measured with a GenRad 2510 MicroModal Analyzer [9],[11]. The transfer function $H(f)$ was then calculated by

$$H(f) = \frac{Sxy(f)}{Sxx(f)} \quad . \tag{1}$$

Therefore, the goal of the project was to produce a parametric model of a system from the non-parametric model, $H(f)$.

A study was made of the available frequency domain identification codes, three of which were chosen as representative of three different approaches taken in solving the problem.

These three were further developed or modified to work as a general Laplace transform transfer function identifier. The codes studied were Transfer Function (TF), developed at LLNL by H. McCue, Transfer (TRANSF), developed by H. J. Weaver, also at LLNL [16], and NAVFIT, originated at Nasa-Ames Research Center by J. Hodgkinson and further developed by M. Tischler [5]. TF identifies the coefficients of a parametric Laplace transfer function model by a linear least-squares method. TRANSF identifies the damping, frequency and residues of a summation of complex pole pairs. Finally, NAVFIT identifies the coefficients of a parametric Laplace transfer function model using a multivariable search method.

In this report, each of the methods is described in detail, and the advantages and disadvantages of each is discussed. User information is included for each code. In addition, some general issues involved in frequency domain identification are discussed.

2.0 TRANSFER FUNCTION

The program Transfer Function (TF) fits frequency response data with an equivalent Laplace transform transfer function model of the system. The system model used is

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots, + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots, + a_0} \quad a_0 = 1 \quad . \tag{2}$$

The parameters identified are the coefficients of the transfer function, $a_1, a_2, \ldots, a_n$, $b_0, b_1, \ldots, b_m$. The method used to solve for the coefficients is a batch linear least-squares algorithm. The derivation of the linear equations is covered in detail in the Appendix of this report.

Weaknesses

The $X^T X$ matrix of the solution (see Appendix) is often ill-conditioned, especially if the frequency range of the data and/or the order of the model is high. This situation occurs when the columns of the X matrix are nearly linearly dependent. When this happens the $X^T X$ matrix is nearly singular, and since the inverse must be computed, the solution may be inaccurate. A more numerically stable method of solving this least-squares problem is the QR decomposition. The QR routines from LINPACK are used in TF. A discussion of the application of the QR decomposition to the linear least-squares problem is contained in the LINPACK User's Guide [14]. The LINPACK routine will automatically improve the condition of the problem by throwing out columns of the X matrix until a set of linearly independent columns are obtained. The coefficients corresponding to the columns that were thrown out are set to 0. Therefore, when any of the high order coefficients have been set to 0 it is an indication that the problem was too ill-conditioned to solve for a model of that order.

Another drawback is that this method tends to identify the higher frequency poles first. Noise in the data often appears to the code as high frequency resonances, so TF will fit the noise before fitting the actual lower frequency poles and zeros. An example of this is shown in figure 1. The solid is data generated by a fourth order system, with random noise added in the frequency domain. The dotted line is the fitting function calculated by TF, given a numerator order of three and a denominator order of four. It is apparent that the algorithm tries to identify the noise at the high end of the spectrum before identifying the actual resonances at the lower end.

This behavior has been observed in time-domain identification codes that use a linear least-squares solution. One approach often used to overcome this problem is to over-model the system [13]. By increasing the model order, both the actual system response and the noise are fitted. figure 2. shows a much imporoved fit of the same data when TF is given a numerator order of 7 and a denominator order of 8. This approach is discussed further in section 5 of this report.

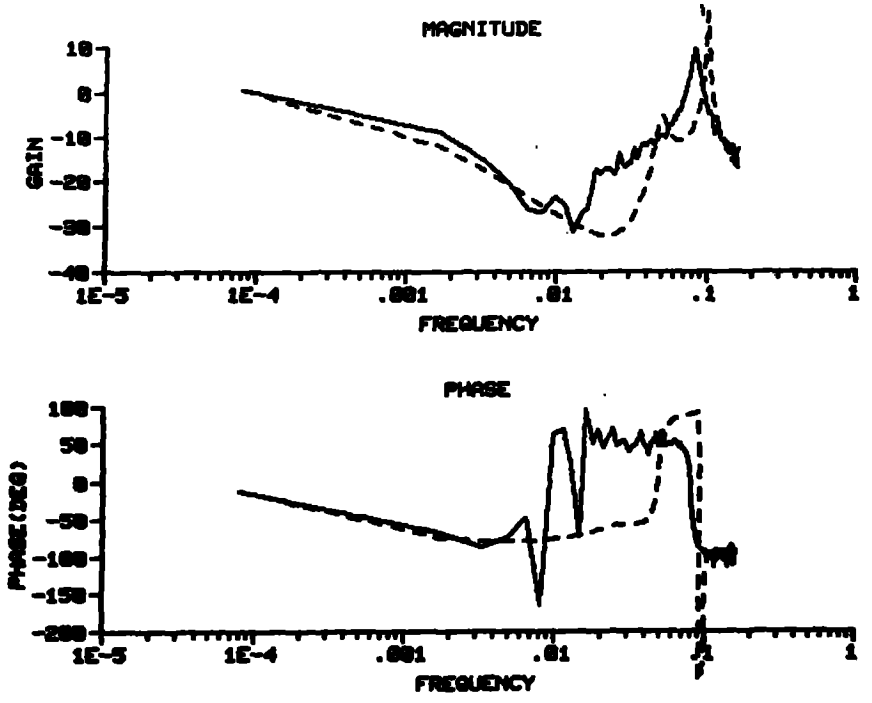

Figure 1. Fourth order system.

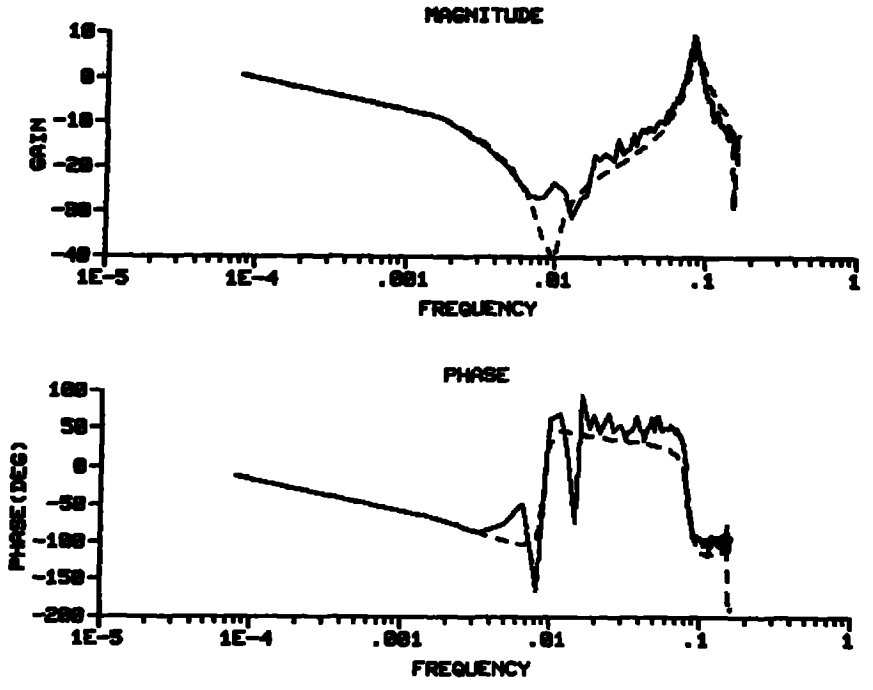

Figure 2. Fourth order system, over-modelled.

Strengths

This is the simplest solution of the problem. It is a one-step solution that will always converge, unlike the non-linear methods that at times may converge on a local minimum, or may not converge at all. Furthermore, it is a simple programming problem, and generally requires much less computer time to arrive at a solution than the iterative non-linear techniques. Finally, TF is robust and easy to use. The model order is the only information necessary to fit the data. A method used to determine the model order when it is unknown is discussed in section 5 of this report.

User Information

TF has been incorporated into SIG (a General Purpose Signal Processing Program) and will be available in SIG as a frequency domain transfer function identifier. For more information on SIG please see the SIG User's Manual [12]. The program requires as input two data files, either the Fourier transforms of the input and output time records, $X(w)$ and $Y(w)$, or for the stochastic case, the autospectrum of the input, $Sxx(w)$, and the cross-spectrum of the input and output, $Sxy(w)$. Also, the user will need to make an estimate of the numerator order, denominator order, and system type (i.e., number of poles at the origin).

The output of the program consists of the coefficients of the transfer function, a plot, in magnitude and phase, of the data overlayed by the fitting function, and the mean squared error between the data and the fit. The coefficients are saved in a SIG coefficient data store. The user is then free to do what he/she wishes with the transfer function coefficients.

3.0 TRANSFER

The code Transfer (TRANSF) was originally developed at LLNL for identification of modal parameters from transfer function data obtained by dynamic structures testing. The model used in this code is

$$H(w) = \sum_{k=1}^{n} \frac{A_k + jB_k}{\sigma_k + 2\pi j\,(w - w_k)} \quad , \tag{3}$$

or, as a Laplace transform,

$$H(s) = \sum_{k=1}^{n} \left[\frac{A_k - jB_k}{s + \sigma_k - j2w_k} + \frac{A_k + jB_k}{s + \sigma_k + j2\pi w_k} \right] \quad , \tag{4}$$

where

$n =$ the number of complex pole pairs

A and B are the real and imaginary parts of the residue of the k^{th} pole

σ_k is the damping ratio and,

w_k is the resonance frequency of pole k.

TRANSF does a linear least-squares fit to a function by a method of linearizing the fitting function. The cost function of this method is

$$J = \sum_{i=1}^{N} \left[(H_D(a)_i - H_F(a)_i)^2 \right] \quad , \tag{5}$$

where

a $= [A_1\ B_1\ w_1\ \sigma_1, \ldots, A_n\ B_n\ w_n\ \sigma_n]$, $n =$ number of poles

$H_D(a)_i$ = the transfer function data,

$H_F(a)_i$ = the fitting function, and

N = number of points.

The model, or fitting function is linearized in the change in the parameters δa, by a first order Taylor series expansion. Then δa is iteratively solved for by linear least-squares until a chosen minimum value of the cost function is reached. The algorithm is described in detail in Bevington [3]. For a derivation of the fitting model and information about how to run the program, see the TRANSF User's Manual [16].

Weaknesses

The model used in this program is restrictive. It is an all-pole model, and assumes that the poles are complex pairs. The code fits only these poles and their residues. In effect this method fits the partial fraction expansion of the transfer function. By recombining the parts, the frequency and damping of the zeros can be found. In general though, the code has difficulty identifying simple poles or zeros that lie along the real axis of a real-imaginary pole plot (as opposed to complex pairs) or poles and zeros at the origin. In structural applications, for which this code was developed, this model works satisfactorily. The data generally consists of series of lightly damped resonances, but this model may not be appropriate for other types of systems in other applications.

A single real pole can be identified by fixing the frequency parameter at a very small number (say .0001, setting it to 0 causes a divide by zero.), then iterating for the residue and damping parameters. For example, figure 3 shows the TRANSF fit of a first order low pass filter with a cutoff frequency of 10 Hz. The frequency parameter was fixed at .001 Hz. After six iterations the fit is so close that the dotted line is difficult to distinguish from the actual data.

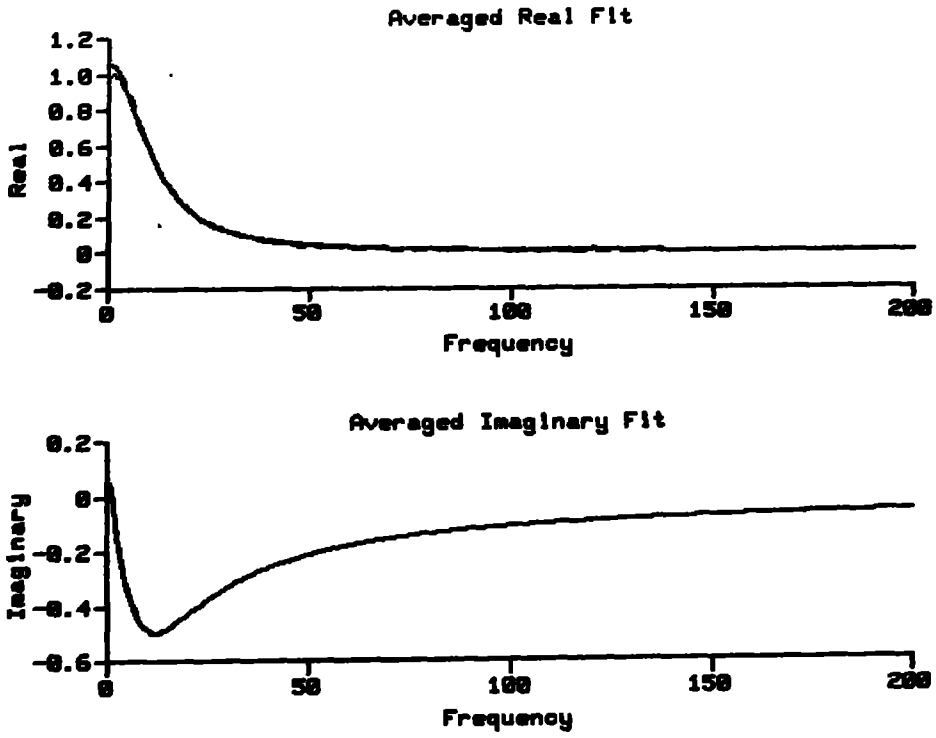

Figure 3. First order low pass filter.

For most data sets a fairly good equivalent transfer function fit can be found (as a combination of complex poles) but if the actual form of the transfer function is known to have simple poles or zeros then those parameters cannot be fitted satisfactorily. For example, a partial fraction expansion of a transfer function that has two simple poles and one simple zero is

$$H(s) = \frac{-149.22 + j8.02e-6}{s+941.91} + \frac{65.69}{s+62.81} + \frac{321.39 + j826.8}{s+314.23+j6276.75} + \frac{321.39 - j826.8}{s+314.23-j6276.7}$$

The best fit produced by TRANSF is shown in figure 4. The fitting function from the identified coefficients is

$$H(s) = \frac{56.21 + j33.77}{S+113.19+j628.32} + \frac{56.21 - j33.77}{s+113.19-j628.32} + \frac{321.39 + j824.88}{s+314.31+j6276.0} + \frac{321.39 - j824.88}{s+314.31-j6276.0}.$$

As shown in the plot, TRANSF can find a good equivalent fit. The complex pole pair at the high end of the spectrum was very accurately identified as can be seen by comparing the last terms in both the actual function and the identified function. However, instead of identifying the simple poles and zero of the actual transfer function, TRANSF fitted that part of the data with an equivalent complex pole pair.

Strengths

The all-pole model used in this method is the model most often used in the frequency domain identification codes studied. For many applications the zeros are of no real interest. If only the location and damping ratio of the poles are needed, as in the structural applications, or if the data is a series of lightly damped resonances, a good fit of the transfer function can be obtained. For instance, in the previous example the complex pair of poles was identified quite accurately. Also, using this method and model, the user has the option to fit one section of the data at a time. If the data record contains several resonant peaks over a wide range of frequencies, one can split up the data record into a few sections and identify one or two peaks at a time. This option can help with numerical problems sometimes encountered with this code when fitting data over a wide range of frequencies.

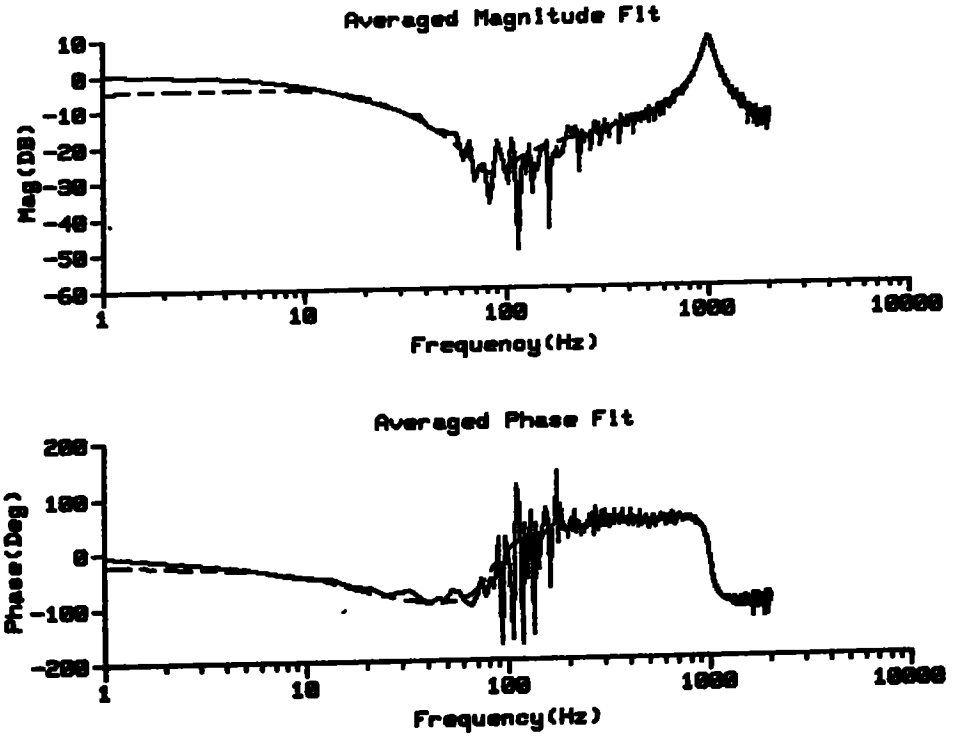

Figure 4. Best fit of fourth-order system by TRANSF.

4.0 NAVFIT

The code NAVFIT is a general purpose program to fit high order system frequency response data with a low order transfer function. NAVFIT was originally developed at the Nasa-Ames Research Center for transfer function identification, to be used in aircraft control problems. The system model used in this code is a transfer function of the form

$$H(s) = \frac{\text{gain}\left(x_1 s^n + x_n s^{n-1} + \ldots + x_n\right) e^{-\tau s}}{\left(x_{n+1} s^m + x_{n+2} s^{m-1} + x_{n+3} s^{m-2} + \ldots + x_{n+m-1}\right)} \quad . \tag{6}$$

The parameters identified are

gain,

$x_1, x_2, \ldots, x_{n+m+1}$; the coefficients, and

τ, the time delay.

NAVFIT uses Rosenbrock's Multivariable Search Method to identify the parameters of the transfer function. This method minimizes a cost function by varying the parameters along a set of orthonormal directions until a minimum is reached. At that point a new set of directions is computed and the parameters are varied again. The two steps (computation of directions, and variation of parameters) are repeated until a certain minimum cost is reached, or until no new set of directions can be found to continue the minimization (a local minimum is reached). For a detailed description of the algorithm see Himmelblau [2].

Weaknesses

This method needs a good set of starting parameters in order to successfully minimize the cost and arrive at an accurate estimate of the parameters of the transfer function. Without good starting parameters the algorithm is very likely to reach some local minimum and never converge to a good fit. For example, to identify the fifth order system shown in figure 5, NAVFIT was given starting coefficients all equal to 1.0. The final fit (the dotted line) is not a satsifactory fit of the data.

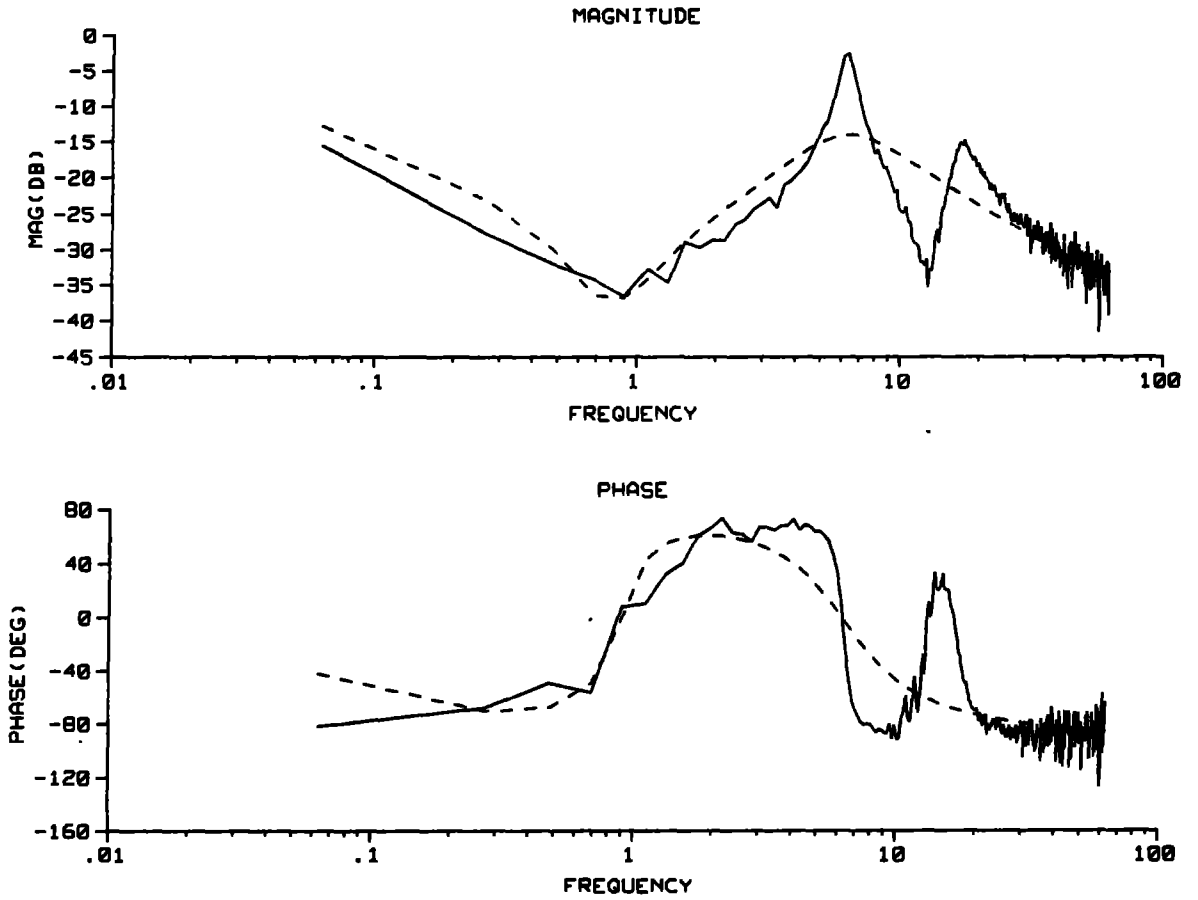

Figure 5. Best fit from poor starting values.

It is necessary to know the order of the numerator and denominator, the type of poles and zeros (simple, or complex pairs) and make a good guess at the resonance frequencies of the poles and zeros. Generally this information can be estimated visually from the Bode magnitude and phase plots of the transfer function data. Some analysis of the physical system can also be helpful. For example the form of the transfer function of any motors, amplifiers, etc. included in the system can help in picking the first guess at model order and resonant frequencies.

Figure 6 shows a good fit of the same data when NAVFIT was given good starting parameters estimated from the Bode plot.

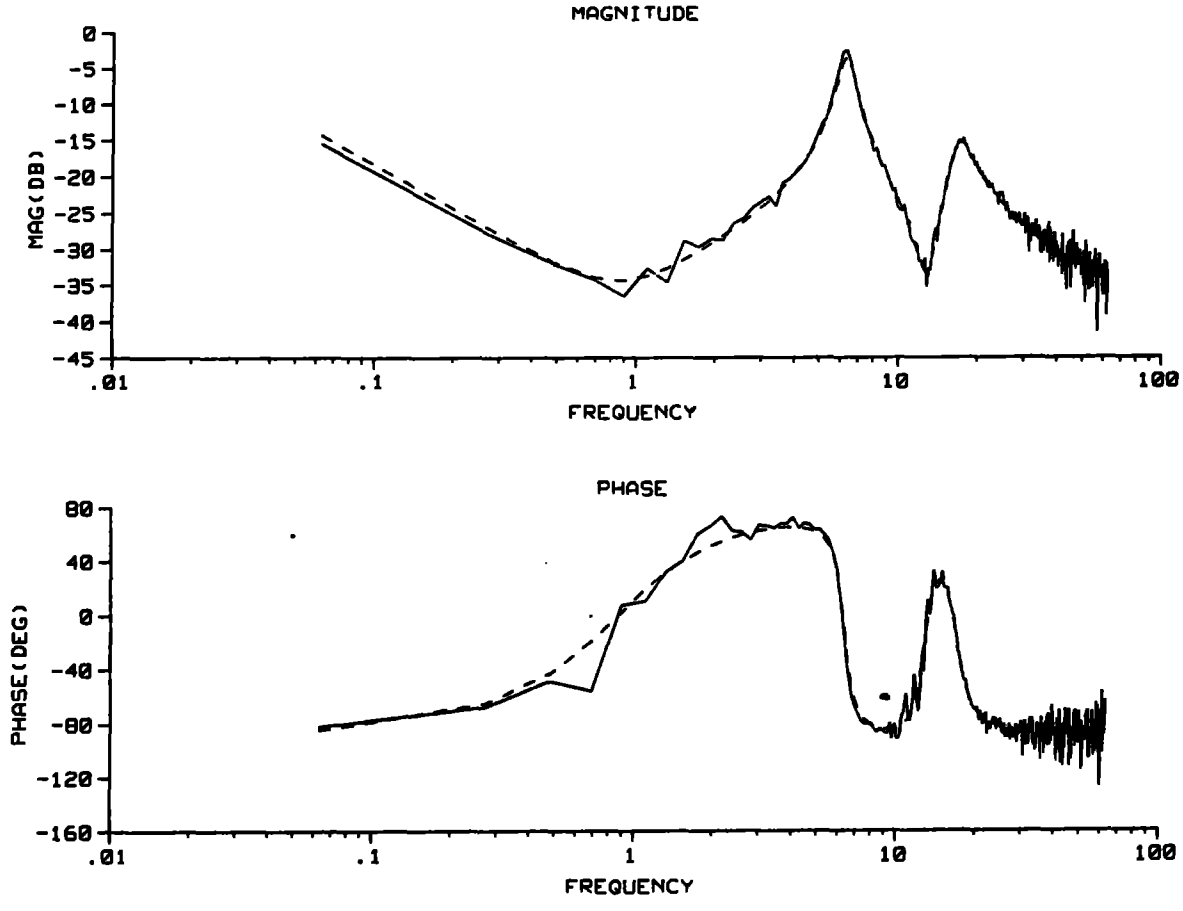

Figure 6. Best fit from good starting values.

This method does a least-squares fit of the data, so the best fit of very noisy data, or data with noise that distorts the shape of the actual response will not necessarily be a correct fit.

Strengths

This is the most general method of the three codes studied. It fits both poles and zeros equally well, and allows for a time delay in the system. A knowledge of the expected form of the system transfer function can be used to your advantage by guiding the program to identify the poles and zeros you expect to see.

Example Run and User Hints

There is no formal User's Manual for NAVFIT at this time, so included here is an example run and detailed user hints. The code is written in FORTRAN and runs on a DEC VAX 11/780 computer. NAVFIT is an interactive program that fits high order response data to a low order transfer function. It will prompt the user for the information needed to start a fit.

The program requires as input a file of transfer function data in triplets of frequency (r/s), real, and imaginary parts of the transfer function data. The user also needs to have a first guess of the order of the numerator and denominator of the transfer function, and starting coefficients. The starting coefficients can be entered as coefficients, or poles and zeros in real-imaginary or damping and frequency format.

The output of the program will be the coefficients of the fitted transfer function, and a plot of the data overlayed by the fitting function. The poles and zeros of the transfer function are calculated and output in both real-imaginary and frequency-damping format. The output information can be printed to the terminal screen and/or saved in a file.

Initially the user must enter the name of the graphics terminal being used, so that the graphics routines will plot correctly. Then the data may be entered at the terminal or read from an input file. Data entered at the terminal is entered in triplets of freq(hz), gain(db) and phase(deg.); one triplet per line.

In this example we use a file containing the high order input response data of the system. The format of the file must be

line 1:	Text, a file identifier of up to 80 characters			
lines 2 - end:	triplets of	Freq(r/s)	real	imaginary

The numbers are read in FORTRAN free format. The limit on the number of data points is 1600, but the more data there is the longer it will take to converge on an answer. It is best use less than 1000 data points if possible.

As an example run, we will use a file called notch.dat, which contains the frequency response of an actual notch filter, 435 points. The user will be prompted as follows. Answers are entered on the following line.

```
*** HIGH ORDER INPUT DATA ***

INPUT TYPE:

(1)  DATA FROM A FILE

(2)  DATA ENTERED AT THE TERMINAL

OPTION NO.?
1
NOTCH.DAT
```

The input data is plotted (figure 7) and the user has the option to re-read the file if there is any problem.

```
INPUT OK [Y/N]?
Y
```

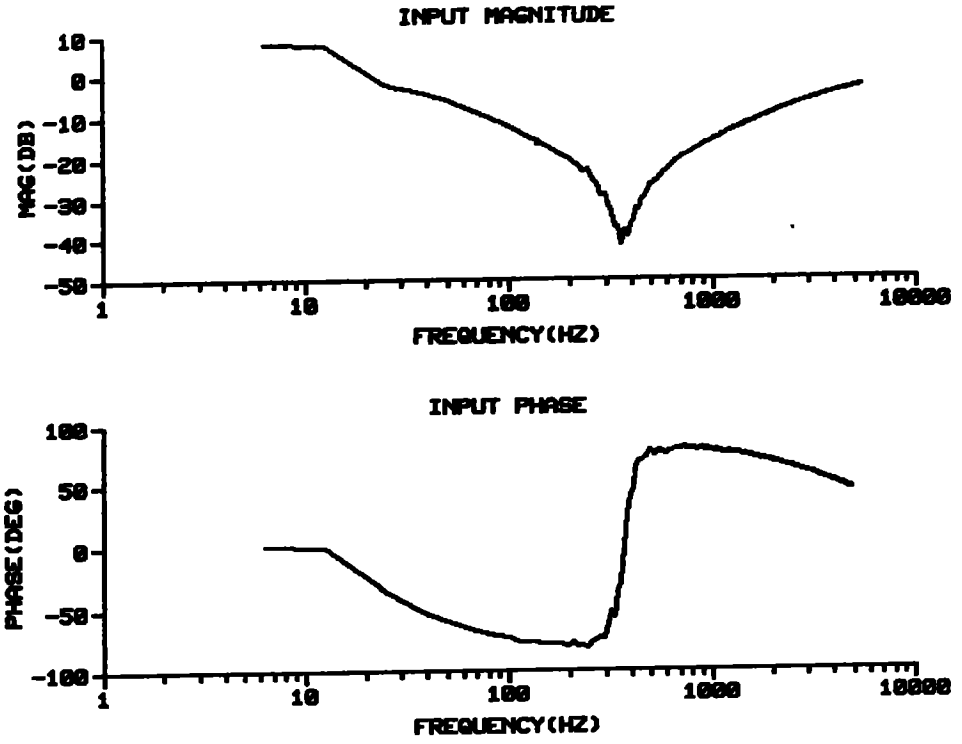

Figure 7. Notch filter data.

Next the user must enter the starting parameters. The order of the numerator and denominator are entered.

```
ENTER 1ST GUESSES, L.O.S. NUM,DENOM ORDERS
2 2
```

Then an options menu is printed to the screen.

```
*** LOW ORDER DATA ***

        (1) TO ENTER COEFFICIENTS

        (2) TO ENTER ROOTS (ZETA,OMEGA OR REAL,IMAG)

        (3) TO ENTER ROOTS (CURSOR PEAKS)

OPTION NO.?
```

The user is given the option to enter (option 1) the starting coefficients, or (option 2 and 3) the roots.

In option 2 the quadratic roots (complex pairs) can be entered in complex (real-imaginary) form or as damping and frequency. In option 3 the damping ratio of all poles and zeros are automatically set to 0.05, and the user picks the resonance frequencies by cursoring the peaks. For this example we choose option 2.

```
OPTION NO.?
2
```

For option 2 the user is prompted for the number of quadratic terms, ie. the number of pairs of complex poles or zeros, in the numerator and denominator.

```
NUMBER OF QUADRATIC TERMS IN NUMERATOR, DENOMINATOR?
1 0
```

Then the values of the first guess poles and zeros are entered. The user may enter them in either damping and frequency, or real and imaginary.

```
ENTER 1 FOR ZETA,OMEGA, 2 FOR REAL,IMAGINARY.
1

ENTER DAMPING AND FREQ. FOR NUMERATOR QUADRATIC NO. 1
0.08 400.0

ENTER (ON ONE LINE) THE SIMPLE DEN. ROOTS
(DONT FORGET THE MINUS SIGN)
-5.0 -1000
```

The frequency data is automatically scaled to between 0 and 10 Hz, and the steady state gain is automatically chosen so that the coefficients are close to the same order of magnitude. The search method converges much faster when the parameters are close in magnitude. The final set of starting parameters are printed at the screen.

The user is then asked if any of the coefficients are to be fixed, ie. not varied in the parameter search. This is useful for instance when there is a pole or zero at the origin, then the last coefficient of the numerator, or of the denominator, can be set at zero.

```
FIX COEFFICIENTS [Y/N]?
N
```

If the answer had been Yes the user would be prompted to enter a Y or an N (yes or no) for each coefficents to be fixed.

The time delay is then entered and fixed or freed. Generally it's a good idea to fix the time delay at 0 and get a best fit, then if there is a time delay in the system the time delay can be freed and the user can see a direct comparison of the effect of the delay. There is no time delay in this system so we will set it to 0.

```
ENTER TIME DELAY
0

FREE TIME DELAY [Y/N]?
N
```

The user also has the option to force all the coefficients to be positive (a necessary condition for a stable system, but not sufficient [15]).

```
ALLOW NEG. COEFFS. [Y/N]?
N
```

In the cost function, the value of the magnitude and phase errors are weighted by separate constants. The defaults are 1.0 for the magnitude and 0.017 for the phase. The user may change the defaults. For instance, if the phase measurements are questionable, then the weighting constant for the phase may be lowered and the fit will then depend mainly on the magnitude.

```
STANDARD WEIGHTS [Y/N]?
Y
```

Finally the user specifies the number of iterations to be used for the search. To see the fit of the starting coefficients enter 1.

```
ENTER NUMBER OF ITERATIONS (TRY 1000). USE MINUS SIGN ON
THE ITERATIONS TO SKIP INTERMEDIATE SEARCH PRINT OUT
1
```

After the iteration is done,

```
COST FUNCTION = 57729.31
ENTER 0 TO GO ON, NEW ITS FOR MORE ITERATIONS.
0
```

The starting coefficients and a plot of the data overlayed with the fit will be plotted on the screen (see figure 8). If the dotted line of the fitting function is near the data and has approximately the same shape, then the code should have no trouble converging on a good set of coefficients.

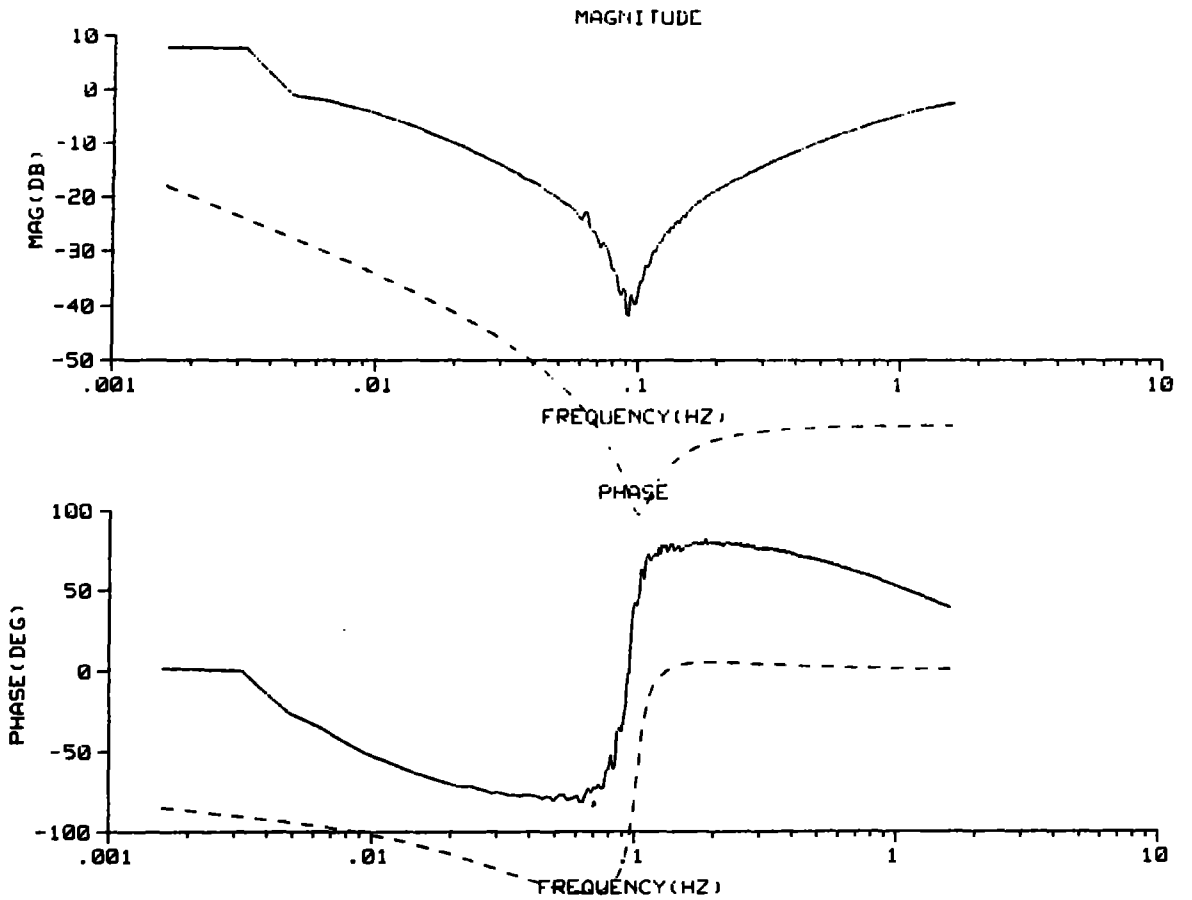

Figure 8. Starting fit of notch filter.

The user is then given several options.

```
********************** NAVFIT OPTIONS ****************************

(1)  NEW INPUT DATA

(2)  NEW 1ST GUESSES, LOW ORDER SYSTEM

(3)  FIX/FREE COEFFICIENTS,MORE ITERATIONS

(4)  DATA OUTPUT

(5)  QUIT

OPTION NO.?
```

If the starting fit looks good, choose (3). The user will be prompted to fix/free coefficients and give a new number of iterations. As it says in the prompt, 1000 iteration is a good number to start with.

If the user wishes to try different starting values, choose (2). The starting orders and coefficients can all be entered again.

To begin with a new set of data, choose (1).

When the user is satisfied with the fit, the final fitting function and plots can be output to the screen and/or a file using option (4).

When finished choose (5) to quit.

During the search, the intermediate values of the cost function, the iteration number, and the parameters are printed to the screen. The cost function is an absolute measure of the sum of the squares of the difference of the magnitude and phase of the data and the fit. With noise in the data the value of the cost will never reach 0. The code will have reached a best possible fit when the message

```
BASE VECTOR NOT CHANGING WITHIN 0.1 PERCENT
```

is printed at the screen. If the code has reached this point during its iterations, yet the plotted fit is still way off, it has reached some local minimum of the cost. To get a better fit, closer starting coefficients will be necessary.

The final fit of the example notch filter data is show in figure 9. The final transfer function calculated is

```
PAYOFF FUNCTION= .657E+01  SIGMA GAIN, PHASE: .409E+00  .304E+01
GAIN(HF)= 0.988225E+00; S.S.= 0.163047E+01; DELAY: 0.000000E+00

     NUMERATOR                    DENOMENATOR
       0.852506830E-07 S** 2        0.862664891E-07 S** 2
       0.279455762E-04 S** 1        0.282882061E-02 S** 1
       0.452428073E+00 S** 0        0.277482808E+00 S** 0
            NUMERATOR                      DENOMENATOR
       REAL           IMAG            REAL           IMAG
 1  -0.16390E+03   0.22979E+04     -0.98387E+02   0.00000E+00
 2     (Z= 0.71147E-01 , W= 0.23037E+04)     -0.32693E+05   0.00000E+00
```

Sigma gain and sigma phase are the mean squared error of gain and phase fits. Gain(hf) is the high frequency gain and S.S. is the steady state gain. The Payoff Function is the value of the cost function for that set of coefficients.

A comparison of the poles and zeros identified by NAVFIT with the actual poles and zeros calulated from the resistor and capacitor values follows. The error is within the accuracy of the capacitor and resistor values. The identified values may well be more accurate than the calculated pole and zero values.

	Actual	Identified	%Error
poles	$5061\,Hz$	$5200\,Hz$	3%
	$25\,Hz$	$15.6\,Hz$	38%
zeros	$w = 360\,Hz$	$w = 366\,Hz$	1%
	$z = 0.0714$	$z = 0.0712$	0.2%

Code Limitations

NAVFIT is limited to 1600 data points, a numerator order of 25, and a denominator order of 25. The data points do not have to be equally spaced along the frequency axis, so the user may edit the data files to delete excess data, or extra noisy points.

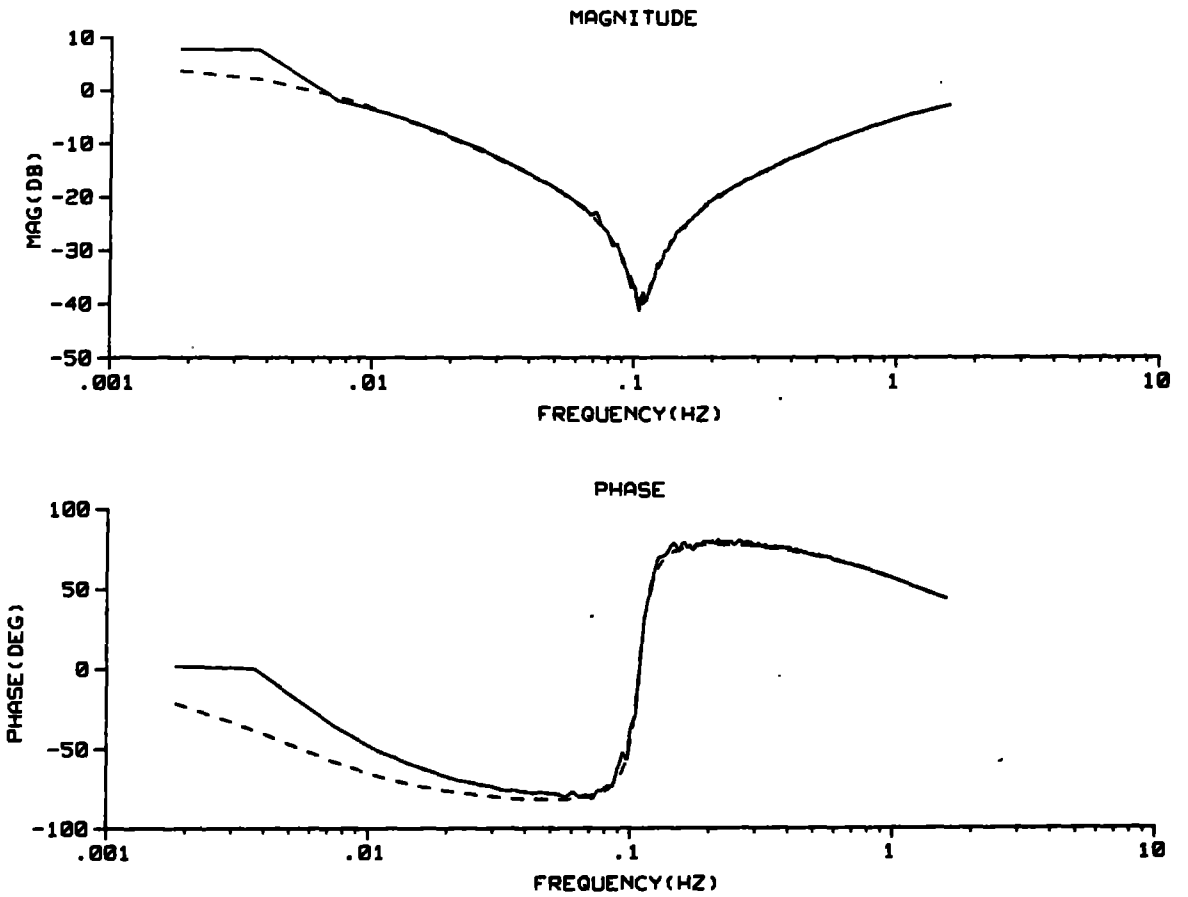

Figure 9. Final fit of notch filter.

5.0 GENERAL ISSUES

There are several issues that should be considered when choosing and running any parameter identification program. Those factors that have an effect on all of the codes considered in this study include sampling of the data - in time and frequency domain, frequency range or the system bandwidth, noise, and model order determination. Each one of these issues and their effects on the various parameter identification codes are discussed next.

Sampled Data

Many data acquisition and analysis systems today take a time record of input and output data and immediately transform it to frequency domain data using an FFT algorithm. The only data available to the engineer then is the Fourier transforms of the input and output data. For this case, the original time data must be sampled correctly according to the Nyquist criterion [8], so that all of the information within the frequency range of interest is preserved, without aliasing the signal. Many of these analyzers today will automatically filter the data and choose the correct sampling rate to meet the Nyquist criterion. The actual data used in this study was collected with a GENRAD 2510 Micromodal Analyzer, using ISAP, the signal analysis program included with the machine [11]. When given a frequency range, number of spectral lines, and a number of averages, the GENRAD will set the cutoff frequency of a programmable low-pass anti-aliasing filter, and choose a sampling rate of 2.56 times the highest frequency. This satisfies the Nyquist sampling criterion. It will then take the number of data records chosen for ensemble averaging, transform them using a micro-coded FFT algorithm, and average them in the frequency domain. The data finally stored and used by the GenRad is the auto-spectra of the input and the output, Sxx and Syy,and the cross-spectrum, Sxy. The transfer function data is calculated by dividing the cross-spectrum by the input auto-spectrum [9].

For the frequency domain data, the choice of sampling rate is not so formalized, but there are some general rules that can be followed for good results. First, the minimum number of points for a transfer function fit must equal the number of parameters being estimated. This guarantees at least one equation for each parameter. Of course, one data point per parameter will rarely be enough data to describe the curve of a more complicated transfer function. In addition, noise in the measured data will distort the transfer function. The goal is to have enough data points to show the general shape of the transfer function, yet not so many points as to waste the computer's and the engineer's time. Some factors to consider when choosing the number of samples are

1. There must be enough points to show the separation of closely spaced poles. The code must be able to identify them separately and identify the zero between them.

2. When the data collected is equally spaced along the linear frequency axis, there must be enough data points at the low end of the frequency spectrum to identify any low frequency dynamics. For instance, a data file may contain 200 data points between .1 to 1000 Hz. This means there are only 2 data points between .1 and 10 Hz. The best fit of those points would be a straight line between them, which is not a good fit if there are any actual resonances under 10 Hz. Either more data points should be measured in that range, or the frequency range should be decreased.

3. When using the non-linear techniques such as NAVFIT, the problem may be viewed as a curve fitting problem. Since NAVFIT does not require equally spaced data, one approach would be to choose many samples around the peaks and valleys of the transfer function plot, and take out those points that do not give much information about the

curve by thinning out the number of points along relatively constant areas of the spectrum.

Frequency Range

Most frequency domain identification algorithms will encounter numerical problems when the freqeuncy range of the data is too great. For instance, in the linear least-squares method, TF, a frequency range of .01 to 1000 Hz with slightly noisy data points, will cause the problem to be too ill-conditioned to obtain an accurate identification of the coefficients. Frequency scaling can center the frequency data about 1, to minimize the chance of underflow or overflow computing errors. Also row scaling of the data matrix can help reduce the range in magnitude of the elements. These scaling techniques have been added to TF, but still a bandwidth over 4 decades can cause numerical inaccuracies.

Similarly, the non-linear programming methods will have difficulty with wide frequency ranges. For a fourth order transfer function with widely spaced poles and zeros, the coefficients of the numerator and denominator can range in size from 1 to 10^{-20}. The non-linear algorithms have trouble converging on a set of parameters with such a wide range of magnitude. Frequency scaling can result in quicker convergence and increased accuracy of the final fitting parameters in most of the parameter identification codes.

Automatic frequency scaling has been added to codes TF and NAVFIT. By scaling the frequency data to between 0 and 1, the coefficients of the transfer function will be scaled to about the same order of magnitude. For example if the transfer function is

$$H(s) = \frac{(2.5e-8)s^3 + (1.4e-4)s^2 + (5.7e-2)s + 3.9}{(4.5e-11)s^4 + (2.1e-8)s^3 + (3.2e-5)s^2 + (9.8e-2)s + 6.5} \quad ,$$

then when scaled by a factor of 1.0e-2 the transfer function coefficients become

$$H(s) = \frac{2.5(1.0e-2s)^3 + 1.4(1.0e-2s)^2 + 5.7(1.0e-2s) + 3.9}{4.5e+1(1.0e-2s)^4 + 2.1(1.0e-2s)^3 + 3.2e-1(1.0e-2s)^2 + 9.8(1.0e-2s) + 6.5}$$

The minimization algorithms of the non-linear programming methods, such as NAVFIT, perform well with these new perameters (the new coefficients). After the code converges on a set of coefficients, the coefficients are scaled back up. Then the rootfinding algorithm factors out and prints the poles and zeros.

Another approach would be to break up the data record into smaller sections, and identify one section at a time. The frequencies within a section can then be limited to a workable range. The code TRANSF has this option included. One point that must be considered when using this approach is that when the data record is split up, dynamics of the system that occur outside the frequencies in the small section can still affect the shape of the transfer function within that section.

To nullify the effect of dynamics outside the frequency range of the section, TRANSF includes a second order bias polynomial. The new fitting model then is

$$H(s) = \sum \left[\frac{A_k + B_k}{s + \sigma_k + 2\pi w_k} + \frac{A_k - B_k}{s + \sigma_k - 2\pi w_k} \right] + (x1)s^2 + (x2)s + x3 \quad . \qquad (7)$$

The coefficients of this polynomial are fitted, along with the parameters of the poles. The final effect is that the curve of the poles is fitted by the pole parameters and the additional curvature from outside dynamics is fitted with the polynomial coefficients, x1, x2, and x3. Therefore, the resonances outside the section frequency range do not effect the estimation of the pole parameters. An example of the effect of the bias polynomial is shown in figures 10 and 11.

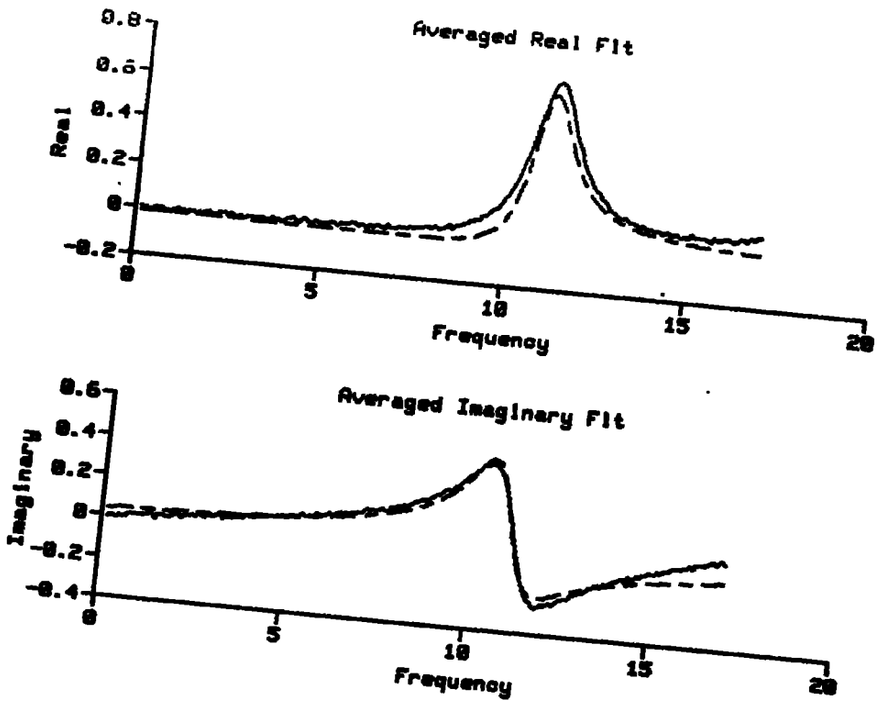

Figure 10. TRANSF fit without bias polynomial.

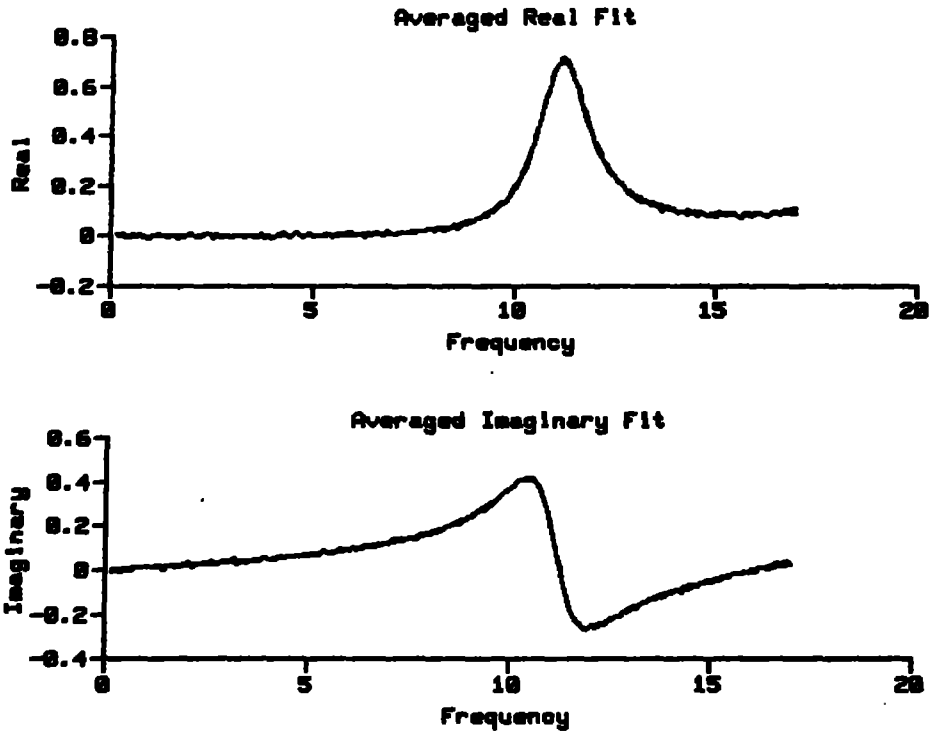

Figure 11. TRANSF fit with bias polynomial.

Since TRANSF fits a summation of pole pairs only, splitting the data between resonance peaks and fitting a few at a time will not affect the final transfer function fit. The data record can be split into sections between peaks, and the final fit can be modelled as a summation of the parameters estimated for each section. This will be a partial fraction expansion of the entire transfer function, so the poles and zeros can be calculated from the recombined transfer function. Still, this method is limited to systems characterized by complex pairs of poles and zeros only.

Noise

The codes using non-linear minimization techniques, TRANSF and NAVFIT both do a least-squares fit to the data. If the noise in the data has a bias, or distorts the shape of the frequency response, then the best least-squares fit is not necessarily the correct fit. When the noise has zero mean or if there is very little noise, the accuracy of the fit can be very high, as is seen in the final fit of the notch filter data (figure 9), described in the previous section on NAVFIT.

The effects of noise on the linear least-squares method are somewhat more complicated. Because the X matrix is often ill-conditioned the error in the data due to noise causes a much greater error in the coefficients estimated. The matrix easily becomes so ill-conditioned that the coefficients calulated are not accurate.

Also, the linear least-squares method tends to identify the high frequency poles and zeros first. Since the noise is seen as high freqency resonances, it will identify the noise before identifying the actual lower frequency resonances. One method often used in practice to overcome this problem in time-domain algorithms, is to over-model the transfer function. For instance, if the system transfer function order is known to be 4, then instruct the program to model it with an order of 10. The extra poles identified will fit the noise. The actual poles and zeros then will have to be picked out by the user. This method has not been tried with TF, but may be worthy of furthur study. An algorithm called REDUCE was developed by D. Goodman at LLNL to pick out the poles that contributed most to the minimization of the error in the least-squares solution of a time-domain identification technique. This is described in the NLS User's Manual [13]. There is also a good description of the over-modelling approach used for a time-domain Prony's identification algorithm in the LLNL report by W. Smith and D. Lager [17]. One of these methods may be useful with TF.

Model Order Determination

On input, all three methods require a model order specification. The best estimation of the model order can be found using a combination of an analysis of the physical system that produced the frequency response data, and judging visually from the Bode plots themselves. The form of the transfer functions of the separate amplifiers, motors, etc. that make up the system can be combined to find the order of the overall transfer function. Also, engineers have been using Bode plots for years to estimate the location of poles and zeros of transfer functions. A resonant peak in the transfer function plot represents a complex pole pair, so a good first guess at model order can be made by counting the peaks and valleys as complex pairs of poles and zeros, the sum of which will give the transfer function denominator and numerator order.

As a last method to estimate if the order is unknown, one can obtain several fits of the data varying the model order. Generally, as the model order increases, the error in the fit decreases, until it levels off at some minimum where a further increase in model order does not decrease the error significantly. The best model order estimate is considered the lowest order that produces an error near that minimum value. This approach may be most useful for the linear least-squares solution, where the best order to fit both the data and the noise is not known.

6.0 CONCLUSIONS AND RECOMMENDATIONS

Since the late 1960's the dominant method used in system identification have been time domain techniques. Though most identification problems can be solved using a time domain algorithm, there are applications where a frequency domain identification technique is the most appropriate. When frequency response methods are used in control systems design, or when the engineer simply wishes to gain general insight into the system, a frequency domain identifier may give the engineer a more intuitive understanding of the system response.

There are several important considerations involved in choosing an identification algorithm, including how much is known about the system that produced the response data. If very little is known about the system, and only a guess can be made about the model order, then a linear least-squares method, such as TF should be used. If on the other hand, the order of the system is known and fairly good starting guesses of the poles and zeros can be made, then the iterative methods such as TRANSF and NAVFIT give the best results.

Another consideration is the model of the system, and its intended use. In structural applications the resonant frequencies and damping ratios are the main concern, so the model used in TRANSF, a summation of complex poles, is best. Other applications, such as control system design, requires identification of the zeros of the system as well as the poles. An identification code that identifies the zeros directly, such as NAVFIT, would be the best choice in this case. In addition, some systems may have simple poles (as opposed to complex pairs, seen in structural systems) or poles or zeros at the origin. The more general identifier NAVFIT is once again the most appropriate choice for these applications.

There are some questions that have not been addressed in this report, and still require further study. Those include:

- What is the best method of distinquishing true poles and zeros from "noise" poles and zeros in an over-modelled response?
- How is the model order best determined?
- Can repeated poles and zeros be identified?
- How do the results of these frequency domain algorithms compare to time domain algorithms?

REFERENCES

1. Forsythe, G. E. and Moler, C. B., *Computer Solutions of Linear Algebraic Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967.

2. Himmelblau, David M., *Applied Nonlinear Programming*, McGraw-Hill, 1972.

3. Bevington, P.R., *Data Reduction and Error Analysis for the Physical Sciences*, McGraw-Hill Book Co., New York, 1969.

4. Ljung, L., "Aspects on the System Identification Problem", *Signal Processing, 4*, 445-456, (1982).

5. Tischler, M.B., Lueng, J., and Dugan, D.C., "Frequency Domain Id. of XV-15 Tilt-Rotor Aircraft Dynamics", *American Institute of Aeronautics and Astronautics*, Report No. AIAA-83-2695, (1983).

6. Goodwin, G.C. and Payne, R.L., *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, New York, 1970.

7. Lawson, C. L. and Hanson, R. J., *Solving Linear Least Squares Problems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1974.

8. Oppenheim, A.V. and Willsky, A.S., *Signals and Systems*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1983.

9. Bendat, J.S. and Piersol, A. G., *Engineering Applications of Correlation and Spectral Analysis*, John Wiley and Sons, Inc., New York, 1980.

10. Ljung, L. and Glover, K., "Frequency Domain Versus Time Domain Methods in System Identification", *Automatica, 17*(1), 71-86, 1981.

11. GenRad, Vibration Analysis Division, *Operating Manual 2510 Micromodal Structural Analyzer*, Santa Clara, CA. 1982.

12. Lager, D. and Azevedo, S., "SIG User's Manual, A General Purpose Signal Processing Program", *Report No. UCID-19912*, Lawrence Livermore Laboratory, November, 1983.

13. Goodnam, D. M., "NLS: A System Identification Package For Transient Signals," *Report No. UCID-19767*, Lawrence Livermore National Laboratory, March, 1983.

14. Dongarra, J.J., Bunch, J.R., Moler, C.B., and Stewart, G.W., "LINPACK User's Guide," *SIAM, 1979.*

15. Dorf, R., *Modern Control Systems*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1981.

16. Weaver, H.J., "TRANSF User Manual," *Report No. UCID-19268*, Lawrence Livermore National Laboratory, November, 1981.

17. Lager, D.L. and Smith, W.D., "Parametric Characterization of Random Processes using Prony's Method," *Report No. UCRL-52673*, Lawrence Livermore National Laboratory, May, 1979.

18. GenRad, Vibration Analysis Division, *ISAP Signal Analysis Program*, Santa Clara, Ca. 1982.

APPENDIX

TF SOLUTION METHOD

TF solves a set of linear equations for the coefficients by using a batch linear least squares solution. The linear equations are derived as follows.

We are given as input data

N = the number of data points
w_i = the discrete frequencies, $i = 1, \ldots, N$
$H(s_i)$ = the transfer function data point at s_i, where $s_i = jw_i$.

The system is modeled then at w_i as

$$H(s_i) = \frac{b_m s_i^m + b_{m-1} s_i^{m-1} + \ldots + b_0}{a_n s_i^n + a_{n-1} s_i^{n-1} + \ldots + a_0} \quad , a_0 = 1 \quad , \tag{A.1}$$

or

$$\left(a_n s_i^n + a_{n-1} s_i^{n-1} + \ldots + a_0\right) H(s_i) = b_m s_i^m + b_{m-1} s_i^{m-1} + \ldots + b_0 \quad ,$$

since $a_0 = 1$

$$H(s_i) = -a_n s_i^n H(s_i) - a_{n-1} s_i^{n-1} H(s_i) - \ldots - a_1 s_i H(s_i) + b_m s_i^m + b_{m-1} s_i^{m-1} + \ldots + b_0 \quad . \tag{A.2}$$

We have one equation of this form for each discrete data point. We can then solve this set of equations

$$H(s) = X\theta \quad , \tag{A.3}$$

where

$$H(s) = [H(s_1), H(s_2), \ldots, H(s_N)]^T \quad ,$$

$$X = \begin{bmatrix} -s_1^n H(s_1) & -s_1^{n-1} H(s_1) & \ldots & -s_1 H(s_1) & s_1^m s_1^{m-1} & \ldots & 1 \\ \vdots & \vdots & & & & & \vdots \\ -s_N^n H(s_N) & -s_N^{n-1} H(s_N) & \ldots & -s_N H(s_N) & s_N^m s_N^{m-1} & \ldots & 1 \end{bmatrix} \quad ,$$

and

$$\theta = [a_1, a_2, \ldots, a_n, b_0, b_1, \ldots, b_m]^T \quad .$$

The solution is of the form

$$\theta = \left(X^T X\right)^{-1} X^T H(s) \quad . \tag{A.4}$$

If we look at $H(s)$ as

$$H(s) = \frac{Sxy(s)}{Sxx(s)} \tag{A.5}$$

or

$$H(s) = \frac{Y(s)}{X(s)} \quad , \tag{A.6}$$

where $s = jw$, then eq. 3 becomes

$$\begin{aligned} Sxy(s_i) = &- a_n s_i^n Sxy(s_i) - a_{n-1} s_i^{n-1} Sxy(s_i) - \ldots - a_1 s_i Sxy(s_i) \\ &+ b_m s_i^m Sxx(s_i) + b_{m-1} s_i^{m-1} Sxx(s_i) + \ldots + b_0 Sxx(s_i) \end{aligned} \quad , \tag{A.7}$$

or, for the stochastic case

$$\begin{aligned} Y(s_i) = &- a_n s_i^n Y(s_i) - a_{n-1} s_i^{n-1} Y(s_i) - \ldots - a_1 s_i Y(s_i) \\ &+ b_m s_i^{m-1} X(s_i) + b_{m-1} s_i^{m-1} X(s_i) + \ldots + b_0 X(s_i) \end{aligned} \quad . \tag{A.8}$$

The $Sxy(s)$ or $Y(s)$, and the corresponding X matrix are substituted in the solution equation (5), so that the solution is of the form

$$\theta = \left(X^T X\right)^{-1} X^T Y(s) \quad , \tag{A.9}$$

or

$$\theta = \left(X^T X\right)^{-1} X^T Sxy(s) \quad . \tag{A.10}$$